IN THE CLAIMS

1. (Currently amended) A method performed by a computer system ofto processing an extensible markup language input stream having a tag structure with start and end tags which may enclose further tags, so that the tag structure may be represented by a tree structure, the method using objects ~~discrete software components~~ mapped to tags contained in the input stream, said objects defining start and end methods which can be invoked, the method comprising:

parsing the input stream,

as a tag is found during the parsing process, building a tree representation of the input stream and the objects bound to tags by building the object mapped to the tag found into the tree representation according to the tag structure,

invoking, in response to finding a start tag, the start method defined in the object mapped to the tag found, and

invoking, in response to finding an end tag, the end method defined in the object mapped to the tag found,

wherein, due to the fact that the object is built into the tree representation before one of the methods defined by it is invoked, the method invoked has awareness of the position of the tag to which it is mapped within the tree structure, as the tree structure is built.

~~parsed, invoking the software component mapped to it.~~

2. (Previously presented) The method of claim 1, wherein the extensible markup language is XML.

3. (Previously presented) The method of claim 1, wherein the discrete software components are classes in an object-oriented programming language or procedures or functions in a procedural programming language.

4. (Currently amended) The method of claim 1~~3~~, wherein the ~~input stream comprises at least one tag which is formed by a start tag and an end tag, and wherein the discrete software components comprise at least one of a method which is invoked as the start tag is parsed (init method) and a~~ end method ~~which~~ is invoked when all children of the tag have been parsed ~~(run method)~~.

5. (Canceled)

6. (Previously presented) The method of claim 1, wherein the mapping between the tags and the discrete software objects is changed before, during, or after the parsing process.

7. (Previously presented) The method of claim 1, wherein an extensible markup language output stream is generated, and the output stream is used as an input stream for another execution of the method.

Al

8. (Previously presented) The method of claim 1, wherein the computer system is a server in a network, and the input stream processed by the server is comprised in a request received from a client over the network or is comprised in an output from a database.

9. (Previously presented) The method of claim 1, wherein the invoked software components comprise at least one software component for accessing a database.

10. (Previously presented) The method of claim 1, wherein the invoked software components comprise at least one of a software component for sending electronic mail and a software component for sending facsimiles.

11. (Currently amended) The method of claim 1, wherein the computer system comprises a local computer and a remote computer which communicate with each other,

and at least some of the commands given to the local computer are executed on the remote computer, but the results of the execution are output on the local computer so that the commands appear to be executed locally, wherein the communication between the local and the remote computers comprises an extensible markup language stream, wherein the method is performed by at least one of the remote computer and the local computer so as to execute the commands or output the results of the execution.

12. (Currently amended) A computer system, comprising:

a processing unit and storage for processing programs,

bindings representing a mapping between tags and methods of discrete software components,

a software engine comprising:

- a reader component arranged to ~~that~~ reads an extensible markup language input stream ~~containing at least one tag~~ having a tag structure with start and end tags which may enclose further tags, so that the tag structure may be represented by a tree structure,

- a parser component arranged to ~~that~~ parses the input stream,

- an execution component ~~that~~, as a tag is found during the parsing process, arranged to: ~~invoke the software component mapped to the tag,~~

build a tree representation of the input stream and the objects bound to tags by building the object mapped to the tag found into the tree representation according to the tag structure,

invoke, in response to finding a start tag, the start method defined in the object mapped to the tag found, and

invoke, in response to finding an end tag, the end method defined in the object mapped to the tag found,

wherein, due to the fact that the object is built into the tree representation before one of the methods defined by it is invoked, the method invoked has awareness of the position of the tag to which it is mapped within the tree structure, as the tree structure is built.

13. (Previously presented) The computer system of claim 12, which comprises two or more of the software engines, wherein at least one of the software engines generates an extensible markup language output stream, and the output stream is used as an input extensible markup language stream for another one of the software engines.

14. (Previously presented) The computer system of claim 12, which is a server in a network, and the extensible markup language input stream processed by the server is comprised in a request received from a user over the network or is comprised in an output from a database.

15. (Previously presented) The computer system of claim 12, which comprises a database, and wherein the invoked software components comprise at least one software component for accessing the database.

a1

16. (Previously presented) The computer system of claim 12, which comprises at least one of an email dispatch system and a facsimile dispatch system, wherein the invoked software components comprise a software component for sending electronic mail or a software component for sending facsimiles.

17. (Previously presented) The computer system of claim 14, which comprises a local computer and a remote computer which communicate with each other, and at least some of the commands given to the local computer are executed on the remote computer, but the results of the execution are output on the local computer so that the commands appear to be executed locally, wherein the communication between the local and the remote computers comprises a extensible markup language stream, and wherein at least one of the remote and the local computers has a software engine so as to execute the commands or output the results of the execution.

18. (Currently amended) A computer program product ~~including~~comprising a machine-readable medium with program code stored on it, for carrying out a method, when executed on a computer system, ~~said program code for~~of processing an extensible markup language input stream having a tag structure with start and end tags which may enclose further tags, so that the tag structure may be represented by a tree structure, the method using objects ~~discrete software components~~ mapped to tags contained in the input stream, said objects defining start and end methods which can be invoked~~, said program code for directing the computer system to~~, the program code being arranged to:

- parse the input stream,
- as a tag is found during the parsing process:

build a tree representation of the input stream and the objects bound to tags by building the object mapped to the tag found into the tree representation according to the tag structure,

a1 invoke, in response to finding a start tag, the start method defined in the object mapped to the tag found,

invoke, in response to finding an end tag, the end method defined in the object mapped to the tag found,

wherein, due to the fact that the object is built into the tree representation before one of the methods defined by it is invoked, the method invoked has awareness of the position of the tag to which it is mapped within the tree structure, as the tree structure is built~~parsed, invoke the software component mapped to it~~.

19. (Canceled)

20. (Previously presented) The computer program product of claim 17, wherein the program code comprises the following classes:

- a class which parses the input stream,
- a class which implements a parser interface,
- a class which creates a document,
- a class which creates a taglet, i.e. which binds objects to tag names,

- a class which provides behavior for the taglets.

21. (New) A signal transmitted over a computer network comprising a representation of program code for carrying out a method, when executed on a computer system, of processing an extensible markup language input stream having a tag structure with start and end tags which may enclose further tags, so that the tag structure may be represented by a tree structure, the method using objects mapped to tags contained in the input stream, said objects defining start and end methods which can be invoked, the program code being arranged to:

- parse the input stream,
- as a tag is found during the parsing process:

build a tree representation of the input stream and the objects bound to tags by building the object mapped to the tag found into the tree representation according to the stream tag structure,

invoke, in response to finding a start tag, the start method defined in the object mapped to the tag found,

invoke, in response to finding an end tag, the end method defined in the object mapped to the tag found,

wherein, due to the fact that the object is built into the tree representation before one of the methods defined by it is invoked, the method invoked has awareness of the position of the tag to which it is mapped within the tree structure, as the tree structure is built.